DOCUMENT RESUME

ED 248 833 IR 011 282

AUTHOR Swigger, Keith
TITLE A Structured Model for Software Documentation.
PUB DATE 21 May 84
NOTE 10p.; Paper presented at the Mid-Year Meeting of the American Society for Information Science (13th, Bloomington, IN, May 21, 1984).
PUB TYPE Viewpoints (120) -- Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Authoring Aids (Programing); Cognitive Processes; *Computer Software; *Design Requirements; *Discourse Analysis; *Guides; Layout (Publications); Programing; *Reading Comprehension; Reading Strategies
IDENTIFIERS *Computer Users

ABSTRACT

The concept of "structured programming" was developed to facilitate software production, but it has not carried over to documentation design. Two concepts of structure are relevant to user documentation for computer programs. The first is based on programming techniques that emphasize decomposition of tasks into discrete modules, while the second was developed in discourse analysis to explain strategies used by readers and to model their cognitive processes in forming mental models of text content. Consideration of the text production and text comprehension approaches together provides a basis for designing "user-friendly" software manuals. A model for structured documentation suggests the need for: modules to be appropriate macropropositions (global content of the text); clear identification of module function as a tutorial, operational, or reference component; planned ordering of modules and explicit superstructures to help readers identify effective strategies; and adequate access points to modules through such devices as indexes. An examination of the surface structures of 15 manuals for microcomputer file management indicated that structural guidance in existing manuals is inadequate. Nine references and the manuals that were examined are listed. (LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED248833

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

A Structured Model for Software Documentation

Presented at the 13th
ASIS Mid-Year Meeting

Bloomington, Indiana
May 21, 1984

Keith Swigger
School of Library Science
Texas Woman's University

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Keith Swigger

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

This paper discusses two instantiations of the concept of "structure" relevant to user documentation for computer programs. The first, following some ideas suggested by Weiss (1982, 1983), is derived from the concept of structure as developed in programming techniques that emphasize decomposition of tasks into discrete modules. The second concept of structure has been developed in discourse analysis to explain strategies used by readers and to model their cognitive processes in forming mental models of content of texts (Pace, 1982, Jonassen, 1982, van Dijk and Kintsch, 1983). By considering these two approaches to structure together, one concerned with text production, the other with text comprehension, some specific suggestions can be made for design of software manuals that will meet demands for "user friendliness."

Weiss suggests we view the relationship between a computer user and a manual as similar to the relationship between a computer and a program. A manual should lead the user through an unambiguous series of steps to accomplish a task, just as a program directs a machine in performance of steps (Weiss, 1983). Given the analogy between a manual and a program, Weiss argues that design and evaluation of manuals can be guided by the same principles that guide structured programming. The writing of manuals, he says, should be structured and quantized. By identifying the information needed for execution of specific tasks, the process of writing can become a process of developing standardized routines. "What we shall have eventually is 'structured documentation,' writing done in small, manageable chunks, each self-contained, each clear, each linked to the rest of the document according to one of a small number of 'standard moves.'" (Weiss, 1983, p. 131)

Weiss's comparison of programs and manuals leads to some practical advice for producing manuals. Units of manuals are

IR011282

understood as tools for satisfying specific information needs. Analysis of information needs defines the components of the manual. Once the required modules are identified, division of labor in writing is possible. Wiess goes so far as to suggest that eventually the production of documentation may be automated: "If we make the modules small enough and logical enough, some day we may even get the machines to write their own documentation." (Weiss, 1983, p. 131) The challenge for a model of documentation production, by human or machine, is identification of modules and of "standard moves."

There are some cautions that must be kept in mind in considering the analogy between a manual and a program, particularly when considering microcomputer software. First, providing instruction on carrying out specific sequences of steps is but one purpose of a software manual, the operations function. Only a few of the activities that manuals instruct users in are operations, where operations include keyboard moves and other equipment manipulation tasks. Operations are of course absolutely essential but they are the least intellectually demanding of computing activities. In large computer systems, where there are clear distinctions between operating personnel, systems personnel, and end users, it makes sense to consider operations manuals as a separate kind of document. Microcomputer software, however, is usually intended for use by a single individual who combines in him/herself all the personnel functions. Documentation for such microcomputer software must serve tutorial functions, operational functions, and reference functions.

The "structured program" view of a manual does not provide sufficient guidance for production of manuals that will serve all three functions: operations, tutorial, and reference. It provides a useful heuristic for operations components of manuals, but the tutorial and reference components of manuals are not intended to tell the user of a system how to do something.

The tutorial function of a manual serves to instruct the user in the principles underlying the solution to a task that has been automated. This function is particularly important in microcomputer software, which may have been selected by a user because he/she has been persuaded that a particular program will accomplish some general purpose ("Knowledge Manager can help you manage your information") (MDBS, 1984). InfoWorld 6:19 (May 7, 1984), 136)) What the user needs to know is how the system works conceptually, and that instruction is the major goal of the tutorial component. In data management software, for example, it is necessary to instruct the user in such concepts as relations or networks. Whereas a computer follows a program without learning from it, people use the tutorial component of documentation to learn concepts which will be retained. The modules in a manual as components of the tutorial process must be seen as units in a teaching design rather than as units describing a process.

The reference component of a manual serves as an aid to memory. Here the commands used in a system are displayed with explanations of the function, and sometimes the syntax, of each. Like the tutorial component, the reference component of a manual provides information rather than guidance on a series of steps to execute. Reference modules must be organized in some apparent fashion (alphabetical order by command or function name is typical) so that they are locateable, but considerations of sequencing may be less crucial than for operations or tutorial modules.

Another difficulty with the analogy between manual and program has to do with the differences between the ways human beings and computers interact with instructions. Computers, of course, follow orders. Human beings are less tractable. In fact, research on human interaction with instructional manuals of various kinds shows that people do not follow the "program." In her summary of research on user behavior with technical manuals outside the realm of computers, Wright (1980, 1983) notes that users tend to exhibit unwillingness to read documentation, at least in the way that books are traditionally organized. Rather, they leaf through or browse in search on answers to specific information needs. It appears from these studies that people recognize a difference between manuals and the traditional linear book, that is, the book that is to be read from beginning to end. Readers do not access technical materials sequentially, so alternative access structures should be provided.

Findings concerning behavior with manuals combined with the multi-functional requirmements for microcomputer program manuals lead to the conclusion that Weiss's argument for modularly structured documentation is too simple as it has been presented so far. Modularity remains a valuable structural principle, but these considerations raise questions concerning the proper ordering and representation of modules, when viewing a text globally, and of the most effective ordering of information within modules viewed locally.

Some insights into possible answers come from research in discourse analysis, particularly van Dijk's and Kintsch's work related to structure of texts. In a series of works (van Dijk and Kintsch, 1977, van Dijk, 1980, van Dijk and Kintsch, 1983), van Dijk and Kintsch have developed a theory of text superstructure, macrostructure and microstructure. Microstructure refers to the ways in which propositions are developed, represented, and understood in individual sentences or paragraphs. Macrostructure refers to the broad representation of a document that forms in a reader's mind as he/she reads the text; studies of macrostructure are concerned with generalizations, or the "gist" of a document that a reader constructs and which is stored in long term memory. Superstructure is "an overall form that organizes the macropropositions (the global content of the text)."(van Dijk and Kintsch, 1983, 16) For many types of text there is a conventional superstructure with which readers are familiar and

which they use as a helping strategy to set up expectations about the order and content of material in texts. Such superstructures are only beginning to emerge for microcomputer documentation. It is reasonable to expect that generic structures will evolve as the literature matures and as writers engaged in mutual imitation. More research is needed on superstructures in various genre, certainly in the genre of computer documentation.

The concepts of macrostructure and superstructure are useful heuristics for modeliing structures for microcomputer documentation, particularly when combined with the notions of modularity proposed by Weiss. Because they are concerned with propositions, summary understandings, and reading strategies, these concepts provide principles on which we can base specific advice for manual writing.

Superstructures may exist for readers before they approach documents. In some genres these are taught in the educational system. We teach a superstructure for research papers, for example, that can be summarized as:

--Statement of the problem
--Significance of the problem
--Review of relevant research
--Description of method
--Discussion of findings
--Limitations of study
--Conclusions

These superstructures are conventional, and thus vary from one discipline or sub-culture to another. In a relatively new genre such as microcomputer documentation, it is not clear whether readers, or specific groups of readers, have learned superstructures or set up structural expectations. Research on that point is needed, but a guess is that these are not well developed. In that case, it is the task of the writer to make the superstructure apparent to the reader through signals in the surface structure of the text.

The theory of acquisition of macrostructures is evolving; in their earlier work, van Dijk and Kintsch argue that macrostructures as orderings of macropropositions are the result of readers' logical processing of propositions in the discourse itself. More recently (van Dijk and Kintsch, 1983) they have suggested that the inference of macrostructures is the result of "contextual macrostrategies" which set up "anticipatory expectations" about topics and of "textual macrostrategies" in which the reader uses properties of the text itself to "provide the definite decisions about the actual topic" (p. 201). Contextual macrostrategies depend upon the knowledge a reader brings to a text; textual macrostrategies depend upon cues provided by the writer.

The goal of a writer, clearly, particularly a technical writer, is to provide cues that will cause the reader to create what is,

from the writer's perspective, a proper macrostructure. As van Dijk and Kintsch observe, "One obvious way is to explicitly tell in the discourse itself what the main topics are" (van Dijk and Kintsch, 1983, p. 202). The devices available for presenting the writer's view of the appropriate macrostructure are familiar ones: underlining topic sentences of paragraphs, use of section headings, chapter titles, tables of contents, and emphatic notes within text. Such surface devices as labels and headings may even be more important in shaping the reader's representation of the text than the semantic processing that takes place in the reader's mind.

The simple implication of this work in discourse analysis is that texts should be replete with guides to the reader. The importance of such surface devices has long been apparent to those involved in indexing and abstracting. These surface signals are precisely the elements of text one turns to to most quickly construct a summary of content. It is interesting to note that in their research on texts, workers in the field of discourse analysis have largely ignored the whole field of indexing and abstracting.

We can synthesize the two major instantiations of structure discussed above to make some suggestions about ways to structure microcomputer software documentation:

--The appropriate modules, in the sense of structured programming, ought to be what the writer sees as appropriate macropropositions. Each macroproposition is a topic, so each ought to be a separate module. To preserve its integrity as a module and to guarantee its perception as a separate topic, it ought to be clearly labeled as such.

--The function of each module as a tutorial, operational, or reference component should be clear to the writer and clearly identified to the reader.

--The search for smallest self-contained modules and for the set of standard moves from module to module must be informed by the differences among operations, tutorial and reference modules. Attention to the differences among types of modules will help prevent confounding of modules.

--The ordering of modules should be planned, but since readers follow strategies rather than algorithms, the ordering should help the reader identify effective strategies. (The effective ordering of propositions remains at the level of art; much more research is needed on which kinds of order are most effective, particularly in documents like microsoftware documentation which must be designed for broad audiences.)

--In the absence of generic superstructures for texts, it is incumbent upon text designers to provide explicit superstructures to help readers formulate reading strategies.

--Since readers use manuals to solve problems (another way

of saying they follow strategies rather than algorithms) and do not read linearly, it is necessary to provide adequate access points to modules, through such devices as indexes.

Much of that advice seems obvious to me as I write it, but apparently it needs to be said because manuals do not follow it. A study of the surface structures of fifteen manuals for microcomputer file management is summarized in Table 1. The purpose of the study was not to identify any particular manual as good or bad but simply to see what structural devices were in use, and with what frequency. No comparative study of user satisfaction, learning rates, or macrostructure acquisition was attempted because the differences in the systems, which vary from the simple PFS File to the complicated dBase II.

TABLE 1. STRUCTURAL SIGNALS IN DATA MANAGEMENT SYSTEM MANUALS

SIGNAL	PRESENT IN MANUAL #:*														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Table of contents	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	(In manual 2, labelled "Index")														
Chapter titles	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Chapter subheadings	*	*	*	*	*	*	*	*	*		*	*	*	*	*
Chapter sub-subheadings	*				*	*							*		*
Labelled page-tabs			*												
Chapter titles in text	*			*	*	*	*	*	*	*	*	*	*	*	*
Chapter subheadings in text	*		*	*	*	*	*	*	*		*	*	*	*	*
Chapter sub-subheadings in text	*				*	*							*		*
Chapter heading on each page				*											*
Italics for definitions						*									
Boldface for emphasis				*							*	*			
Keyword markers (e.g. NOTE,IMPORTANT,etc.)	*	*			*	*	*				*			*	
Additional subheads not in Table of Content	*	*					*	*						*	
Numbered features	*	*									*	*		*	
Numbered steps	*	*													
Subject Index (to page numbers)				*		*	*	*			*		*	*	*
Subject Index (to text subdivisions)													*		

*Manuals are identified by number in the Appendix.

The table is presented to suggest that if the structural considerations discussed are valid, then clearly existing manuals

are inadequate and the complaints commonly voiced about difficulties in use of manuals may be related to inadequate structural guidance. A valid test of the argument would require writing sets of manuals for particular software using different structural devices for manuals for the same system; such a set is currently in preparation for dBase II.

The problem of identifying modules or macrostructural elements should be considered separately from the problem of gaining access to them. If readers are to be guided structurally in their use of manuals, the access tools provided in a manual should consistently direct them to, or at least indicate, the beginnings of modules. In programming terms, users should be directed to the beginning of procedures, not allowed to jump into the middle of them. Tables of contents, page tabs, and chapter headings are examples of structural devices that serve primarily as structural signals and secondarily as access devices.

Indexes, on the other hand, are primarily access devices and secondarily structural signals. The access they provide, however, ought to be to the beginning point of modules. Of the manuals surveyed in Table 1, ten had indexes. Of these, entries in only one point to structural elements such as sub-headings of chapters; this manual uses boldface in the pointers to identify the places in text where the subject identified in the index is explained as a primary proposition. The utility of indexes is a subject of dispute among manual designers, some of whom question its cost effectiveness. Variations in index design (including omission of the index) will be included in construction of the set of test manuals of varying structure described above.

The aim of this paper has been to relate work in two fields, one well understood and tested in practice, the other at the stage of emerging theory. Principles of structured programming are widely accepted as guides in data processing; it is tempting to transfer the techniques to a less-well understood problem, writing. But in doing so, we must also consider theories of knowledge acquisition through reading. The aim of this paper has been to synthesize these two approaches to structure to provide some concrete suggestions for those who must face the practical problem of producing a manual.

REFERENCES

Jonassen, David H. Implicit Structures in Text. In Jonassen, ed. The Technology of Text. Englewood Cliffs, N.J.: Educational Technology Publications, 1982, 5-14.

Pace, Ann Jaffe. Analyzing and Describing the Structure of Text. In David H. Jonassen, ed., The Technology of Text. Englewood Cliffs, N.J.: Educational Technology Publications, 1982, 15-28.

van Dijk, Teun A. Macrostructures: An Interdisciplinary Study of

Global Structures in Discourse, Interaction, and Cognition. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1980.

van Dijk, Teun A. and Walter Kintsch. Cognitive Psychology and Discourse; Recalling and Summarizing Stories. In Wolfgang U. Dressler, ed., Current Trends in Textlinguistics. New York: Walter de Gruyter, 1977, pp. 61-80.

van Dijk, Teun A. and Walter Kintsch. Strategies of Discourse Comprehension. New York: Academic Press, 1983.

Weiss, Edmond H. The Writing System for Engineers and Scientists. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1982.

Weiss, Edmond H. Usability: Toward a Science of User Documentation. ComputerWorld 17 (Jan. 10, 1983), In-Depth, 9-10+.

Wright, Patricia. Usability: The Criterion for Designing Written Information. In Paul A. Kolers, Merald E. Wrolstad, and Herman Bouma, eds., Processing of Visible Language 2. New York: Plenum Press, 1980, pp. 183-205.

Wright, Patricia. Manual Dexterity: A User-Oriented Approach to Creating Computer Documentation. In Proceedings CHI 83 Human Factors in Computing Systems. Boston, Dec. 12-15, 1983. New York: ACM, 1983, pp. 11-18.

APPENDIX
Manuals Examined

1. CONDOR User's Manual. Ann Arbor: Condor Computer Corp., 1982.

2. Data Bank Information Management System. Coral Gables, FL: Data Access Corp., 1982.

3. The Data Factory. Highland Park, Ill.: Micro Lab, 1981.

4. Datafax. Santa Monica: Link Systems, 1982.

5. Datakeyper. Hollis, N.H.: ESP Computer Resources, Inc., 1982.

6. dBase II User Manual. Culver City, CA: Ashton-Tate, 1982.

7. DB Master. San Rafael, CA: Stoneware Microcomputer Products, 1980.

8. The Executive Secretary User's Manual. John Risken, 1981.

9. File Manager+. Richmond, CA: Synapse Software, 1981.

10. JINSAM Training Guide. Riverdale, N.J.: Jini Micro Systems,

Inc., 1982.

11. PFS. Mountain View, CA: Software Publishing Corp., 1981.

12. R:BASE Tutorial. Bellevue, WA: Microrim, Inc., 1983.

13. RL-1 User's Manual. Ann Arbor: ABW Corp., 1982.

14. Versaform User's Guide. Los Gatos, CA: Applied Software Technology, 1981.

15. Visifile User's Guide. Sunnyvale, CA: Personal Software, Inc., 1981.